TCAM Razor: A Systematic Approach Towards Minimizing Packet Classifiers in TCAMs

Chad R. Meiners Alex X. Liu Eric Torng
{meinersc,alexliu,torng}@cse.msu.edu
**Introduction**

Access Control List

First Match Conflict Resolution

No Modifications

Firewall

Packet Classifier

<table>
<thead>
<tr>
<th>Rule</th>
<th>SIP</th>
<th>DIP</th>
<th>S Port</th>
<th>D Port</th>
<th>Protocol</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.2.3.0/24</td>
<td>192.168.0.1</td>
<td>0</td>
<td>*</td>
<td>*</td>
<td>discard</td>
</tr>
<tr>
<td>2</td>
<td>1.2.3.0/24</td>
<td>192.168.0.1</td>
<td>65535</td>
<td>*</td>
<td>*</td>
<td>discard</td>
</tr>
<tr>
<td>3</td>
<td>1.2.3.0/24</td>
<td>192.168.0.1</td>
<td>*</td>
<td>0</td>
<td>*</td>
<td>discard</td>
</tr>
<tr>
<td>4</td>
<td>1.2.3.0/24</td>
<td>192.168.0.1</td>
<td>*</td>
<td>65535</td>
<td>*</td>
<td>discard</td>
</tr>
<tr>
<td>5</td>
<td>1.2.3.0/24</td>
<td>192.168.0.1</td>
<td>[0,65535]</td>
<td>[0,65535]</td>
<td>TCP</td>
<td>accept</td>
</tr>
<tr>
<td>6</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>discard</td>
</tr>
</tbody>
</table>
How do we solve this problem?

• Software solutions
  – Linear search
  – Trie and geometric data structures

• Hardware solution
  – Ternary Content Addressable Memory
    • TCAM
    • Ternary bit pattern keys (0,1,*)
    • Constant time lookup
TCAM Challenges

• Low capacity
  – Maximum of 18 Mb
  – 2 Mb and 1Mb modules are the most popular
  – Each entry has 144 bits
    • 128K entries, 14K entries, 7K entries

• Larger capacity consumes more energy
  – Power is a concern
  – Cooling is a concern

• Larger capacity consumes space

• Larger capacity is expensive
  – 1Mb module is ~ $250

• Minimizing the number of entries addresses these issues
Encoding Considerations

- Range expansions
  - Encoding ranges has a multiplicative effect
    - \([1,14] \Rightarrow 0001, 001*, 01**, 10**, 110*, 1110\)
    - \([1,14] \land [1,14] \Rightarrow 6 \times 6 = 36\) entries
  - 1 rule can lead to over 900 TCAM entries.
    - Each port interval can leads to 30 rules
    - w-bit interval \(\Rightarrow 2w-2\) maximum entries

- Prefix Encoding
  - Standard practice to encode entries in prefix format
    - Each field is in prefix format
  - Minimizing arbitrary ternary lists is NP-hard
Minimizing Prefix Entries

• Minimal Prefix encodings is polynomial
  – For single fields only!
  – Multiple fields is an open problem
  – Dynamic programming algorithm
    • Suri et al.
From Single to Multiple Dimensions

- Break the problem into sub problem
  - K dimensional problem
    - K-1 dimensional sub-problems
- Firewall Decision Diagrams
  - Provides hierarchy of K-1 sub-problem
  - Each sub-problem is a sub-tree of FDD
Firewall Decision Diagrams

- First step of TCAM Razor is to convert rules into an FDD

<table>
<thead>
<tr>
<th>Rule</th>
<th>(F_1)</th>
<th>(F_2)</th>
<th>Decision</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>[9,9]</td>
<td>[0,15]</td>
<td>d</td>
</tr>
<tr>
<td>2</td>
<td>[8,11]</td>
<td>[0,7]</td>
<td>d</td>
</tr>
<tr>
<td>3</td>
<td>[8,11]</td>
<td>[12,15]</td>
<td>d</td>
</tr>
<tr>
<td>4</td>
<td>[8,11]</td>
<td>[0,15]</td>
<td>a</td>
</tr>
<tr>
<td>5</td>
<td>[0,15]</td>
<td>[0,15]</td>
<td>d</td>
</tr>
</tbody>
</table>
Multi-dimensional TCAM Minimization

• Work from the bottom up

<table>
<thead>
<tr>
<th>Rule</th>
<th>Field</th>
<th>Decision</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10**</td>
<td>accept</td>
</tr>
<tr>
<td>2</td>
<td>****</td>
<td>discard</td>
</tr>
</tbody>
</table>

These decisions are weighted!
Solution Composition

- Sub-solutions are composed by expansion

<table>
<thead>
<tr>
<th>Rule</th>
<th>Field</th>
<th>Decision</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1001</td>
<td>( v_3 )</td>
</tr>
<tr>
<td>2</td>
<td>10**</td>
<td>( v_2 )</td>
</tr>
<tr>
<td>3</td>
<td>****</td>
<td>( v_3 )</td>
</tr>
</tbody>
</table>

\[ v_1 \]

\[ v_2 \]

\[ v_3 \]

<table>
<thead>
<tr>
<th>Rule</th>
<th>Field</th>
<th>Decision</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10**</td>
<td>accept</td>
</tr>
<tr>
<td>2</td>
<td>****</td>
<td>discard</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Rule</th>
<th>( F_1 )</th>
<th>( F_2 )</th>
<th>Decision</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1001</td>
<td>****</td>
<td>discard</td>
</tr>
<tr>
<td>2</td>
<td>10**</td>
<td>10**</td>
<td>accept</td>
</tr>
<tr>
<td>3</td>
<td>10**</td>
<td>****</td>
<td>discard</td>
</tr>
<tr>
<td>4</td>
<td>****</td>
<td>****</td>
<td>discard</td>
</tr>
</tbody>
</table>
Redundancy Removal

- Apply known redundant rule removal algorithm
- Prefix encoding enables effective redundant rule removal

<table>
<thead>
<tr>
<th>Rule</th>
<th>$F_1$</th>
<th>$F_2$</th>
<th>Decision</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1001</td>
<td>****</td>
<td>discard</td>
</tr>
<tr>
<td>2</td>
<td>10**</td>
<td>10**</td>
<td>accept</td>
</tr>
<tr>
<td>3</td>
<td>10**</td>
<td>****</td>
<td>discard</td>
</tr>
<tr>
<td>4</td>
<td>****</td>
<td>****</td>
<td>discard</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Rule</th>
<th>$F_1$</th>
<th>$F_2$</th>
<th>Decision</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1001</td>
<td>****</td>
<td>discard</td>
</tr>
<tr>
<td>2</td>
<td>10**</td>
<td>10**</td>
<td>accept</td>
</tr>
<tr>
<td>3</td>
<td>10**</td>
<td>****</td>
<td>discard</td>
</tr>
</tbody>
</table>
Experimental Data

• Real-life Packet Classifiers
  – 17 structurally distinct classifiers
  – 42 actual classifiers
  – A few rules to hundreds of rules

• Synthetic Packet Classifiers
  – Difficult to get real-life packet classifiers
  – 18 sets of 100 uniformly sized classifiers
    • Randomly generated
    • Generated to look like real classifiers
Experimental Metrics

• Direct Rule Expansion $\text{Direct}(f)$
  – Number rule produced by encoding classifier $f$ into prefix format
  – We compare against direct rule expansion

• Algorithm application $\text{A}(f)$
  – Number of prefix rules produced applying algorithm $\text{A}$ on classifier $f$
    • TCAM Razor
    • Redundancy Removal
Experimental Metrics

• For a set of classifiers $S$
  – Average Compression ratio of $A$ over $S$
    • $\sum_{f \in S} (A(f)/\text{Direct}(f)) / |S|$
  – Total Compression ratio of $A$ over $S$
    • $\frac{\sum_{f \in S} (A(f))}{\sum_{f \in S} (\text{Direct}(f))}$
  – Average Expansion ratio of $A$ over $S$
    • $\frac{\sum_{f \in S} (A(f)/|f|)}{|S|}$
  – Total Expansion ratio of $A$ over $S$
    • $\frac{\sum_{f \in S} (A(f))}{\sum_{f \in S} (|f|)}$
Experimental Factors

- Field Ordering
  - FDD field order results in a substantial difference
  - $5! = 120$ permutations
- Fortunately there are good permutations
Compression Ratios

- TCAM Razor: 18.2% average compress ratio
- Redundancy Removal: 41.8%
Compression Ratios

- 13 of 17 classifier have less than 1%
Expansion Ratio

The diagram illustrates the percentage of classifier groups across different expansion ratio ranges. The expansion ratio is categorized into five ranges:

1. [0, 0.5]
2. (0.5, 1]
3. (1, 25]
4. (25, 50]
5. (50, 200]

The expansion ratio is plotted on the x-axis, while the percentage of classifier groups is plotted on the y-axis. The expansion ratio is represented as follows:

- TCAM Razor (49)
- Redundancy Removal
- Direct TCAM Expansion

The bars show the distribution of classifier groups across these categories, with each range having a specific pattern and value range.
Synthetic Packet Classifiers

- Average compression ratio
  - .046
- Average expansion ratio
  - 8.737
Synthetic Packet Classifiers

- Total compression ratio
  - 0.016
- Total expansion ratio
  - 3.082
Concluding Remarks

• TCAM Razor is not **optimal**
  – Better algorithms are future work
• TCAM can usually result in significant space savings
  – Average 82% reduction in TCAM entries
• No hardware modification
  – Can be used to improve existing hardware
Questions?